A Supplementary Material

A.1 Problem formalisation

In this section, we provide the MILP formulation of both use cases of our experimental evaluation.

Resource-Constraint Project Scheduling Problem We base our formalisation on the single mode variant of the RCPSP problem in [23]. We consider a project with J activities to complete. W.o.l.o.g. activity 1 is the only start activity, and activity J is the only finish activity. Each activity j has a given processing time d_j, and a set of R_j resources which j needs to be completed. Moreover, each activity j has a set of activity predecessors $\mathcal{P}_j$, which implies that activity j can only start when each of its predecessors $h \in \mathcal{P}_j$ has been completed. The planning horizon is divided into T time units labelled as $t = 1, \dots, T$, where T is a makespan upper bound. Given the precedence relations and the activity duration, we can calculate the time windows, i.e., intervals $[EF_j, LF_j]$, with the earliest and latest finishing times of each activity. In addition, we consider that in each time t there are K_r units of resource $r \in R$ available. A given activity j requires an amount k_{jr} of resource r per period. Given those elements, the goal is to find a feasible schedule that minimises the total duration, i.e., minimises the finishing time of activity J.

We present the formalisation of the RCPSP as follows.

$$\text{minimise} \quad \Phi(x) = \sum_{t=EF_J}^{LF_J} t \cdot x_{Jt} \tag{5a}$$

$$\text{s.t.} \sum_{t=EF_j}^{LF_j} x_{jt} = 1,\ j = 1, \dots, J, \tag{5b}$$

$$\sum_{t=EF_h}^{LF_h} t \cdot x_{ht} \leq \sum_{t=EF_j}^{LF_j} (t - d_j) \cdot x_{jt},\ j = 2, \dots, J,\ h \in \mathcal{P}_j, \tag{5c}$$

$$\sum_{j=1}^{J} k_{jr} \sum_{q=\max(t,EF_j)}^{\min(t+d_j-1,LF_j)} x_{jq} \leq K_r,\ r \in R,\ t = 1, \dots, \bar{T}, \tag{5d}$$

$$x_{jt} \in \{0,1\},\ j = 1, \dots, J, \quad t = EF_j, \dots, LF_j \tag{5e}$$

where x_{jt} are the decision variables expressing whether an activity j is completed on time t. The goal of RCPSP (5a) is to minimise the finishing time of the last activity J. Constraint (5b) forces to schedule each activity once. Constraint (5c) ensures the precedence relations. Constraint (5d) limits the use of renewable resources.

Winner Determination Problem We consider the WSP for the WDP formalisation. There are a set of goods G and a set of bids B. Each bid $b \in B$ is

Table 4: Set of possible queries, the query constraints and the constraint encoding of the WDP formalisation.

	Question	**Query constraints**	**Constraint encoding,** C_Q
1.	Why is bid b selected?	**veto** b	$x_b = 0$
2.	Why is bid b **not** selected?	**enforce** b	$x_b = 1$
3.	Why is bid b selected instead of bid b'?	**enforce** b', **veto** b	$x_{b'} = 1, x_b = 0$
4.	Why is good g selected?	**veto** g	$\sum_{b \in B \mid g \in G_b} x_b = 0$
5.	Why is good g **not** selected?	**enforce** g	$\sum_{b \in B \mid g \in G_b} x_b = 1$
6.	Why is good g selected instead of good g'?	**enforce** g', **veto** g	$\sum_{b \in B \mid g' \in G_b} x_b = 1$, $\sum_{b \in B \mid g \in G_b} x_b = 0$
7.	Why are goods g and g' in the same bid?	**veto** g and g' in the same selected bid	$\sum_{b \in B \mid g', g \in G_b} x_b = 0$
8.	Why are goods g and g' **not** in the same bid?	**enforce** g and g' in the same bid	$\sum_{b \in B \mid g', g \in G_b} x_b = 1$

a tuple $\langle w_b, G_b \rangle$, where w_b is the value of the bid and $G_b \subseteq G$ is the set of the goods within the bid. We formalise the WSP as follows.

$$\text{maximise} \quad \Phi(x) = \sum_{b \in B} w_b \cdot x_b \tag{6a}$$

$$\text{s.t.} \sum_{b \in B \mid g \in G_b} x_b \leq 1, \qquad g \in G, \tag{6b}$$

$$x_b \in \{0,1\}, \qquad b \in B, \tag{6c}$$

where x_b are the decision variables expressing whether a bid b is selected. The goal of WSP (6a) is to maximise the weighted sum of the bids. Constraint (6b) ensures that selected bids are pairwise disjoint.

A.2 Query Translation

In Table 4 we show possible questions tailored to the WDP.

A.3 IIS example

In Table 5, we provide the mathematical encoding of the constraints within the IIS of the running example in the left table of Figure 2.

A.4 Theorem 1 proof

Theorem 1 Given an IIS, the dual graph of an IIS D is connected.

Table 5: *IIS* from Example 3. We indicate the constraint type (in different colours) and the mathematical encoding of the constraints.

Id	Type	Constraint Encoding
q	Query	$\sum_{t=1}^{40} x_{24,t} = 1$
c_1	Completion	$\sum_{t=EF_{16}}^{LF_{16}} x_{16,t} = 1$
c_2	Completion	$\sum_{t=EF_{17}}^{LF_{17}} x_{17,t} = 1$
c_3	Precedence	$\sum_{t=EF_{16}}^{LF_{16}} t \cdot x_{16,t} \leq \sum_{t=EF_{22}}^{LF_{22}} (t - d_{22}) \cdot x_{22,t}$
c_4	Precedence	$\sum_{t=EF_{17}}^{LF_{17}} t \cdot x_{17,t} \leq \sum_{t=EF_{22}}^{LF_{22}} (t - d_{22}) \cdot x_{22,t}$
c_5	Precedence	$\sum_{t=EF_{22}}^{LF_{22}} t \cdot x_{22,t} \leq \sum_{t=EF_{23}}^{LF_{23}} (t - d_{23}) \cdot x_{23,t}$
c_6	Precedence	$\sum_{t=EF_{23}}^{LF_{23}} t \cdot x_{23,t} \leq \sum_{t=EF_{24}}^{LF_{24}} (t - d_{24}) \cdot x_{24,t}$
c_7	Resource	$\sum_{j=1}^{J} k_{j,4} \sum_{q=\max(23,EF_j)}^{\max(23+d_j-1,LF_j)} x_{jq} \leq K_4$

Proof. We prove by contradiction that a disconnected IIS dual graph, i.e., a graph with multiple disconnected components, is not possible to obtain due to the infeasible and irreducible properties of an IIS (Definition 3).

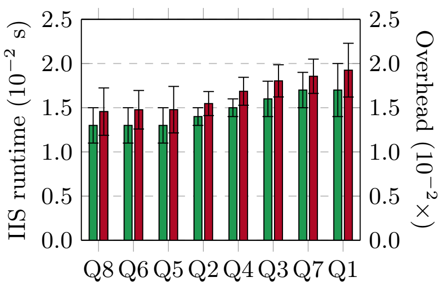
Consider an IIS graph that is disconnected in two subgraphs, $A = (C_A, E_A)$ and $B = (C_B, E_B)$ such that

$$C_A \cup C_B = IIS,$$
$$C_A \cap C_B = \emptyset,$$
$$S_A \cap S_B = \emptyset,$$

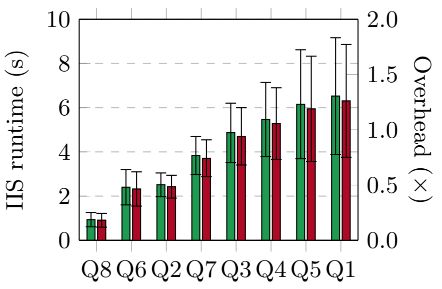
where S_A, S_B are the scopes of the set of constraints C_A and C_B respectively. Notice that, since the graph is disconnected, A and B subgraphs do not share variables, i.e., $S_A \cap S_B = \emptyset$. Then, since the subset of constraints C_A and C_B are (disjoint) subsets of the IIS, they are feasible (see Definition 3). However, since the both subsets of constraints have different variables (different domains), the union of both constraint sets would not affect the feasibility status of the resulting system of constraints. As a result, we would have a feasible IIS, which is a contradiction in itself. Therefore, it is not possible to have a disconnected IIS graph.

A.5 Extended WDP results

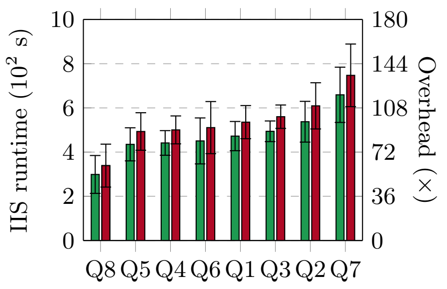
Figures 7 and 8 show the average running time and overhead for each query type. We plot the overhead with "$\times$" units, indicating the cost of computing the IIS compared to the time to solve a problem instance. We organise the results in 6 plots with two distributions: *matching* (Fig. 7) and *paths* (Fig. 8); and three instance sizes based on the number of bids $|B|$: small ($|B| \in \{20, 50\}$), medium



(a) Small



(b) Medium

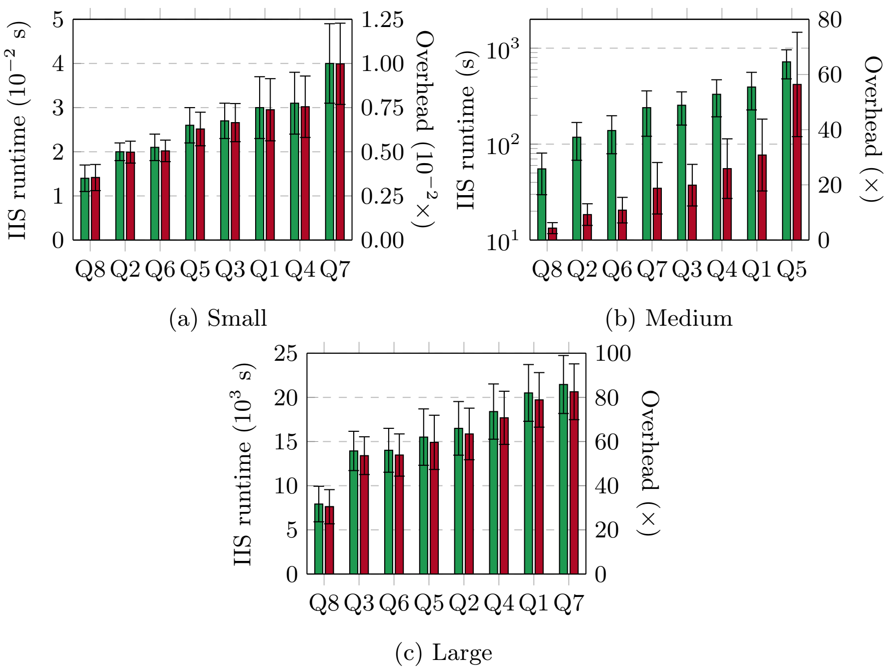


(c) Large

Fig. 7: Average IIS Runtime (green) and Overhead (red) for different query types for the WDP and distribution *matching*.

($|B| \in \{100, 200\}$), and large ($|B| \in \{500, 1000\}$). In general, we observe that as instance size increases, average running times and overheads increase too. In addition, we notice that the order of query types according to empirical hardness varies depending on the instance size and the distribution used to create the instance.

Query type 8 is the easiest query to compute both for the *paths* and *matching* distributions, while queries 7, 1, and 5 tend to be the most difficult. In detail, we show that for small-size instances, computing explanations is much cheaper than solving the problem, i.e., the overhead $\ll 1\times$, and fast —not more than $2 \cdot 10^{-2}$ and $4 \cdot 10^{-2}$ seconds on average for the *matching* and *paths* distributions respectively—. For medium-size *matching* instances (Fig. 8b), the IIS is computed in 6 seconds at most and the overhead increases up to 1.25×. However, for the medium-size *paths* instances (Fig. 8b), the overhead increases significantly —the overhead is up to 55× the solving time for the hardest queries— as well as the runtime —the runtime is between 1 and 10 min on average—. Finally, computing the IIS is hard for large instances since the overhead increases for both distributions. However, the results for the large-size instances (Fig. 7c and 8c) indicate that *matching* instances are easier to explain than *paths* instances since *matching* instances' IISs are computed in minutes while *paths* instances' IISs are computed in hours.



(a) Small

(b) Medium

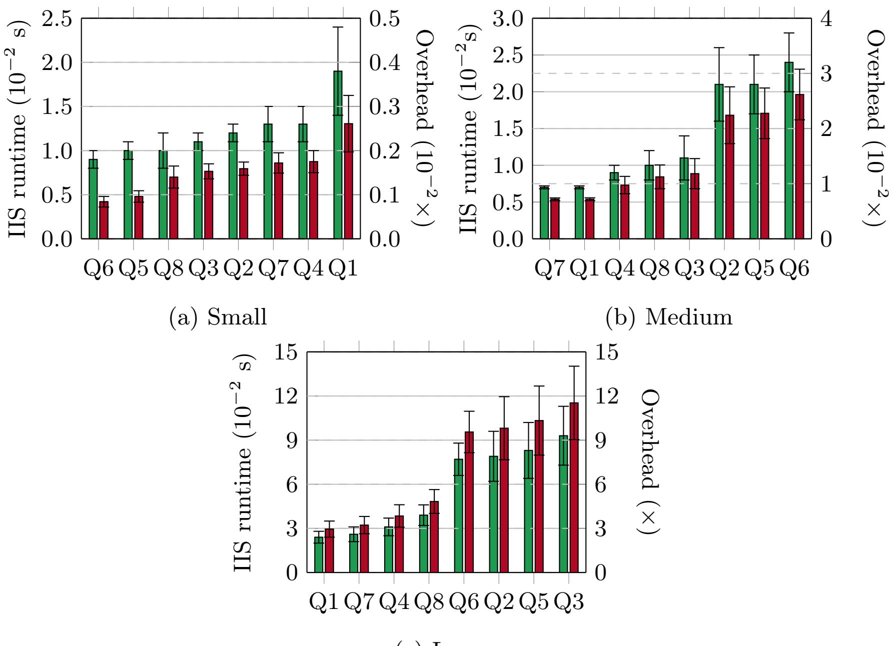
(c) Large

Fig. 8: Average IIS Runtime (green) and Overhead (red) for different query types for the WDP and distribution *paths*.

Figures 9 and 10 show the results of the *scheduling* and *regions* distributions respectively. Similar to Figures 7 and 8, we organise the results in 6 plots with two distributions and three instance sizes: small ($|B| \in \{20, 50\}$), medium ($|B| \in \{100, 200\}$), and large ($|B| \in \{500, 1000\}$).

Results for the instances generated with *regions* distribution show a similar behaviour to distribution *paths*. Moreover, results for the instances generated with *scheduling* distribution show that their explanations can be computed very fast (< 0.1 seconds) for any size.

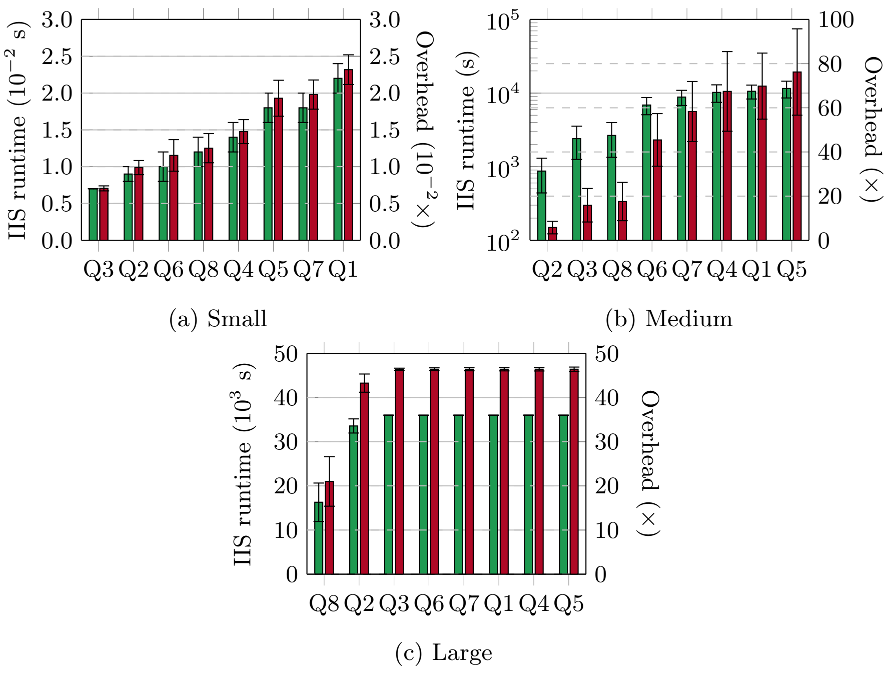
These differences in the IIS runtime confirm the findings of previous works [26] since researchers have reported large differences in the empirical hardness of solving instances from different distributions.



(a) Small (b) Medium

(c) Large

Fig. 9: Average IIS Runtime (green) and Overhead (red) for different query types for the WDP and distribution *scheduling*.



(a) Small

(b) Medium

(c) Large

Fig. 10: Average IIS Runtime (green) and Overhead (red) for different query types for the WDP and distribution *regions*.